A General Purpose Sparse Matrix Parallel Solvers Package†

Hong Q. Ding* Robert D. Ferraro*

Abstract

A general purpose solver package for constructing and solving a range of sparse linear systems arising from discretization of PDEs on unstructured meshes is developed. Once the sparse symmetric complex matrix is constructed, it can be solved by either a preconditioned bi-conjugate gradient solver, a two-stage Cholesky LDL^T factorization solver, or a hybrid solver combining the above two methods.

1 Introduction

Discretizing PDEs on unstructured meshes often leads to linear systems with sparse coefficient matrices. For a large class of problems, the sparse matrix is symmetric, but is often complex as in electromagnetic. For these class of sparse matrix system, few analytical results are known, and their solutions are largely by trying several different methods. Typically, one first tries preconditioned hi-conjugate gradient (PBCG) to see if it converges. If not, one may try a Cholesky factorization LDL^T (not the LL^T for positive definite matrices), which is more robust and stable than the PBCG, although taking longer CPU time and more memory.

For these reasons, we developed a solver package, which provide these different solution methods with a unified user interface. Once the user sets up the geometry (mesh) data and edge information (the matrix elements in the sparse matrix), he may try to solve the system using the method best suited to his problem, or switch to another for comparison.

2 Domain Decomposition

A unique feature of the solver is that, from users viewpoint, they are dealing with a local mesh on each processor, and the sparse matrix construction/assembly proceeds exactly as on a sequential computer. Connecting local patches into a global mesh and thus local matrices into a global one is handled by the package. This is achieved by using a unique geometric approach [l] (see also [2]) in domain decomposition, instead of the more common *algebraic approach* where columns (or rows) of the sparse matrix are distributed to a processor[2].

Subdomains of the unstructured grids are divided among processors. The subdomain boundaries always cut through edges. Grid points sitting on the domain (processor) boundaries are shared among the processors. A matrix element representing an edge on the subdomain boundary between processors p_1, p_2 is split into two pieces $A_{ij} = A_{ij}^{(p_1)} + A_{ij}^{(p_2)}$ where the finite element on processor p_1 contributes to $A_{ij}^{(p_1)}$ and the finite element, on processor p_2 contributes to $A_{ij}^{(p_2)}$; no communication is needed to assemble the matrix element $A_{ij}^{(p_1\ or\ p_2)}$. In this way, calculation of the edge information is entirely based on the local mesh. This significantly cases the programming effort for the user.

† Work funded by the NASA HPCCESS Project.

*Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109.

3 Preconditioned Bi-Conjugate Gradient

PBCG involves two kinds of communications. One is involved in the dot product of two vectors, which is easily done using a global sum of the partial clot product. The other is the matrix-vector multiplication. This involves fairly sophisticated communications within the row based (or column based etc) decompositions. But in our decomposition, this is rather straight forward, involving a local matrix multiplication and an inter-processor communication called globalize(). First, since all entries arc local] y stored, the local $V_1 =$ M. *V.* proceeds as just like on a sequential computer. Afterwards, the only entry in V_1 which arc not complete arc those shared boundary points, which can be completed by summing up all contributions from the several processors which share this point.

At present, only diagonal preconditioning is implemented. Incomplete Cholesky preconditioners arc not implemented, because their usefulness for the symmetric complex matrices has not been documented. Only the nonzero elements of the sparse matrices arc stored.

4 Two-stage Cholesky Factorization

Our domain decomposition naturally classifies the grids into interior grids and boundary grids. This division leads to a well structured pattern of sparsity which allows our Cholesky factorization method to proceed in two phases. Starting with the global equation $\begin{pmatrix} K_{ii} & \\ K_{bi} & \end{pmatrix} \begin{pmatrix} \cdot & \cdot \\ \cdot & \cdot \end{pmatrix} = \begin{pmatrix} f_i \\ f_b \end{pmatrix}$ where K_{ii} stands fOr interior coupled to interior, and K_{bb} for boundary coupled to boundary, etc. Since each block in K_{ii} reside entirely on one processor, we can do a complete local LDL^T factorization t o obtain its inverse, thus obtaining $(K_{bb} - K_{bi}K_{ii}^{-1}K_{ib})x_b = f_b - K_{bi}K_{ii}^{-1}f_i$ or simply $\bar{K}_{bb}x_b = \bar{f}_b$ This reduced equation is much smaller because it deals with only boundary grids. $\bar{K}_{bb}$ is stored as a variable-banded (Skyline profile) matrix, and the equation is solved using a parallel column-based Cholesky factorization, much like a dense matrix. Currently, $K_{bi}K_{ii}^{-1}K_{ib}$ is calculated using a column approach; this scales as $N_i^2 N_b$ on a processor and dominates the CPU time, although it remains a constant as the problem size increase with a increasing number of processors. Taking fully the advantage of the sparsity of K_{ib} will reduce this to N_b^3.

We can also solve the reduced equation by the PBCG, instead of using the parallel Cholesky factorization. This leads a two-stage hybrid solver.

5 Performance

We have completed the solvers package. The package is applied both to a finite element solution of electromagnetic wave scattering from conducting sphere, and to a finite difference solution of a heat distribution problem. We measured the scaling behavior of the solver for the heat distribution problem by increasing the problem size N proportional to number of processors while fixing 1600 grid points per processor. The PBCG method scales as N, and the two-stage methods scale as $\sqrt{N}$; both of them are expected theoretically.

References

[1] G. A. Lyzenga, A. Raefsky and B. Nour-Omid, *Implement Finite Element Software on Hypercube Machines, in* The Third Conference on Hypercube Concurrent Computers and Applacations, Ed. G.C. Fox, p.1755, 1988, ACM Press, New York.

[2] M. T. Heath, and P. Raghavan, *Performance of a Fully Parallel Sparse Solver*, in Proceedings of Scalable High Performance Computing Conference 1994, p.334, IEEE Computer Society Press, Los Alamitos, CA.